HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face

- **Sources:** 37th Conference on Neural Information Processing Systems (NeurIPS 2023)
- **Authors:** Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, Yueting Zhuang
- **Citation:** 514
- **Implementation:** https://github.com/microsoft/JARVIS

Saman Khamesian **03/19/2024**

INTRODUCTION

- Solving complicated AI tasks with different domains and modalities is a key step toward artificial general intelligence.
- While there are numerous AI models available for various domains and modalities, they cannot handle complicated tasks autonomously.
- Despite the successes of current large language models like ChatGPT, current LLM technologies are still imperfect and confront some urgent challenges on the way to building an advanced system.
- The authors discuss them from these aspects:

INTRODUCTION

1) Limited to the input and output forms of text generation, current LLMs lack the ability to process complex information such as vision and speech, regardless of their significant achievements in NLP.

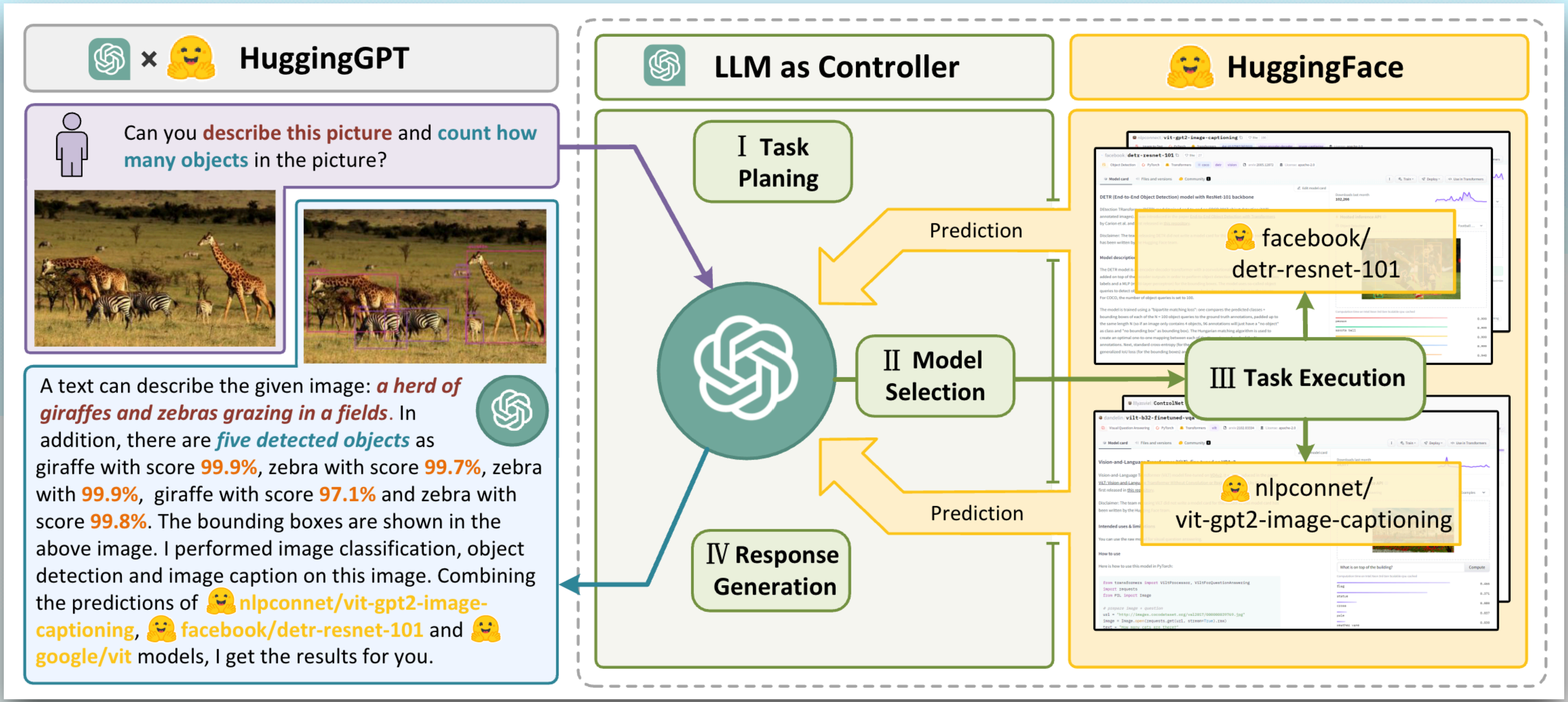
2) In real-world scenarios, some complex tasks are usually composed of multiple sub-tasks, and thus require the scheduling and cooperation of multiple models, which are also beyond the capability of language models.

3) For some challenging tasks, LLMs demonstrate excellent results in zero-shot or few-shot settings, but they are still weaker than some experts (e.g., fine-tuned models).

MAIN IDEA

- In this paper, they propose an LLM-powered agent named HuggingGPT to autonomously tackle a wide range of complex AI tasks, which connects LLMs (i.e., ChatGPT) and the ML community (i.e., Hugging Face) and can process inputs from different modalities.
- On one hand, this model disassembles tasks based on user requests, and on the other hand, assigns suitable models to the tasks according to the model description.
- By executing models and integrating results in the planned tasks, HuggingGPT can autonomously fulfill complex user requests.

MAIN IDEA



× HuggingGPT
Can you describe this picture and count how many objects in the picture?
A text can describe the given image: a herd of giraffes and zebras grazing in a fields. In addition, there are five detected objects as giraffe with score 99.9%, zebra with score 99.7%, zebra with 99.9%, giraffe with score 97.1% and zebra with score 99.8%. The bounding boxes are shown in the above image. I performed image classification, object detection and image caption on this image. Combining the predictions of nlpconnet/vit-gpt2-image-captioning, facebook/detr-resnet-101 and google/vit models, I get the results for you.
LLM as Controller
HuggingFace
Ⅰ Task Planing
Ⅱ Model Selection
Ⅲ Task Execution
Ⅳ Response Generation
Prediction
Prediction
facebook/detr-resnet-101
nlpconnet/vit-gpt2-image-captioning

In this concept, an LLM acts as a controller, managing and organizing the cooperation of expert models. The LLM first plans a list of tasks based on the user request and then assigns expert models to each task. After the experts execute the tasks, the LLM collects the results and responds to the user.

MAIN IDEA

The whole process can be divided into four stages:

1) **Task Planning:** Using ChatGPT to analyze and understand the requests of users, and disassemble them into possible solvable tasks.

2) **Model Selection:** To solve the planned tasks, ChatGPT selects expert models that are hosted on Hugging Face based on model descriptions.

3) **Task Execution:** Invoke and execute each selected model.

4) **Response Generation:** Finally, ChatGPT is utilized to integrate the predictions from all models and generate responses for users.

TASK PLANNING

- They formulate task planing as the first stage of HuggingGPT, which aims to use LLM to analyze the user request and then decompose it into a collection of structured tasks.

- They require the LLM to determine dependencies and execution orders for these decomposed tasks, to build their connections.

- To enhance the efficacy of task planning in LLMs, HuggingGPT employs a prompt design, which consists of:

1) **Specification-based instruction**

2) **Demonstration-based parsing**

SPECIFICATION-BASED INSTRUCTION

❖ The AI assistant performs task parsing on user input, generating a list of tasks with the following format:

```
[{“task”: task, “id”: task_id, “dep”: dependency_task_ids, “args”:
{“text”: text, “image”: URL, “audio”: URL, “video”: URL}}]
```

`“task”:` It represents the type of the parsed task. It covers different tasks in language, visual, video, audio, etc.

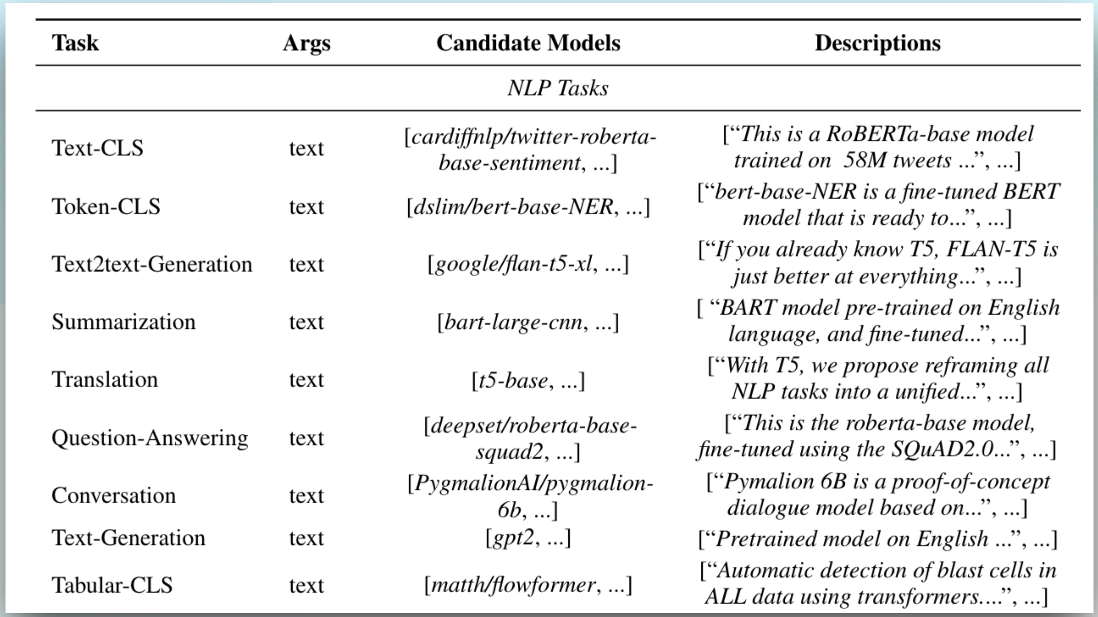
`“id”:` The unique identifier for task planning, which is used for references to dependent tasks and their generated resources.

`“dep”:` It defines the pre-requisite tasks required for execution. The task will be launched only when all the pre-requisite dependent tasks are finished

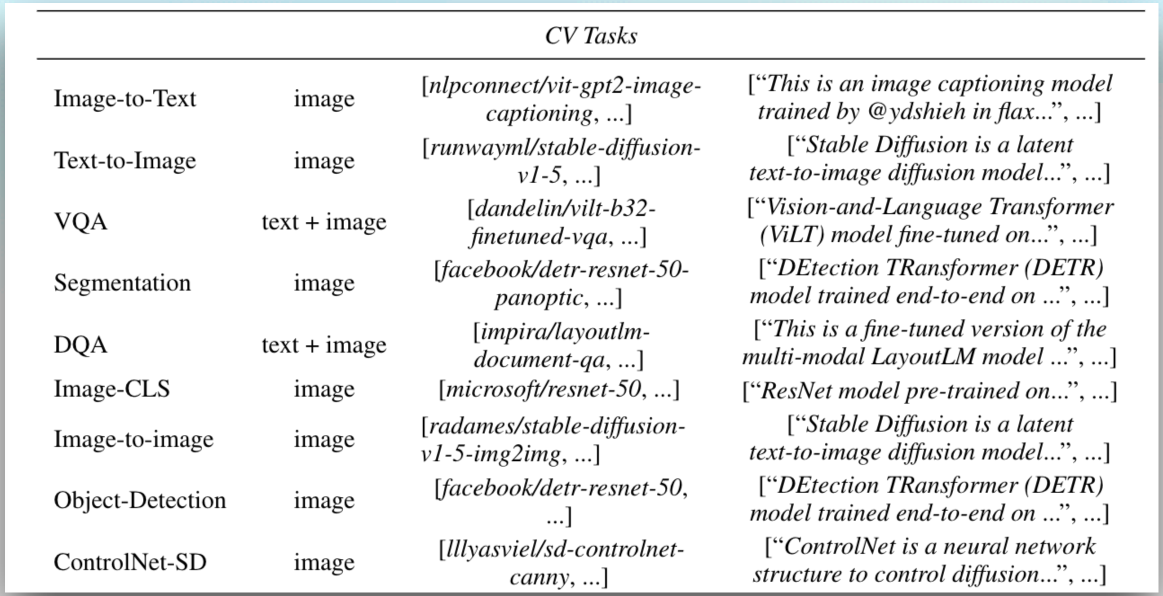
`“args”:` It contains the list of required arguments for task execution.

SPECIFICATION-BASED INSTRUCTION

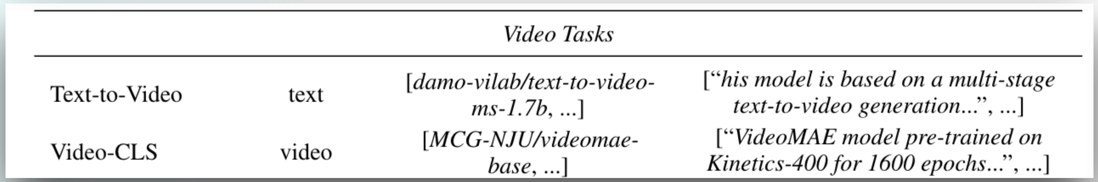
- The task must be selected from the following options:



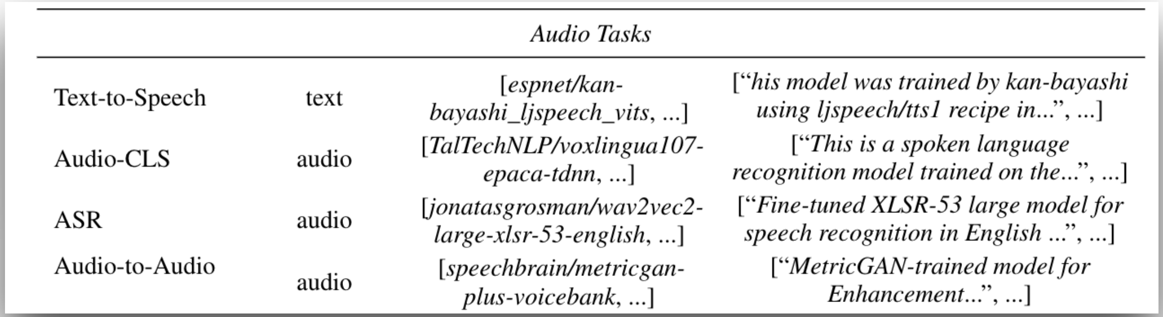
Task	Args	Candidate Models	Descriptions
		NLP Tasks	
Text-CLS	text	*[cardiffnlp/twitter-roberta-base-sentiment, ...]*	*["This is a RoBERTa-base model trained on 58M tweets ...", ...]*
Token-CLS	text	*[dslim/bert-base-NER, ...]*	*["bert-base-NER is a fine-tuned BERT model that is ready to...", ...]*
Text2text-Generation	text	*[google/flan-t5-xl, ...]*	*["If you already know T5, FLAN-T5 is just better at everything...", ...]*
Summarization	text	*[bart-large-cnn, ...]*	*["BART model pre-trained on English language, and fine-tuned...", ...]*
Translation	text	*[t5-base, ...]*	*["With T5, we propose reframing all NLP tasks into a unified...", ...]*
Question-Answering	text	*[deepset/roberta-base-squad2, ...]*	*["This is the roberta-base model, fine-tuned using the SQuAD2.0...", ...]*
Conversation	text	*[PygmalionAI/pygmalion-6b, ...]*	*["Pymalion 6B is a proof-of-concept dialogue model based on...", ...]*
Text-Generation	text	*[gpt2, ...]*	*["Pretrained model on English ...", ...]*
Tabular-CLS	text	*[matth/flowformer, ...]*	*["Automatic detection of blast cells in ALL data using transformers....", ...]*



		CV Tasks	
Image-to-Text	image	*[nlpconnect/vit-gpt2-image-captioning, ...]*	*["This is an image captioning model trained by @ydshieh in flax...", ...]*
Text-to-Image	image	*[runwayml/stable-diffusion-v1-5, ...]*	*["Stable Diffusion is a latent text-to-image diffusion model...", ...]*
VQA	text + image	*[dandelin/vilt-b32-finetuned-vqa, ...]*	*["Vision-and-Language Transformer (ViLT) model fine-tuned on...", ...]*
Segmentation	image	*[facebook/detr-resnet-50-panoptic, ...]*	*["DEtection TRansformer (DETR) model trained end-to-end on ...", ...]*
DQA	text + image	*[impira/layoutlm-document-qa, ...]*	*["This is a fine-tuned version of the multi-modal LayoutLM model ...", ...]*
Image-CLS	image	*[microsoft/resnet-50, ...]*	*["ResNet model pre-trained on...", ...]*
Image-to-image	image	*[radames/stable-diffusion-v1-5-img2img, ...]*	*["Stable Diffusion is a latent text-to-image diffusion model...", ...]*
Object-Detection	image	*[facebook/detr-resnet-50, ...]*	*["DEtection TRansformer (DETR) model trained end-to-end on ...", ...]*
ControlNet-SD	image	*[lllyasviel/sd-controlnet-canny, ...]*	*["ControlNet is a neural network structure to control diffusion...", ...]*



		Video Tasks	
Text-to-Video	text	*[damo-vilab/text-to-video-ms-1.7b, ...]*	*["his model is based on a multi-stage text-to-video generation...", ...]*
Video-CLS	video	*[MCG-NJU/videomae-base, ...]*	*["VideoMAE model pre-trained on Kinetics-400 for 1600 epochs...", ...]*



		Audio Tasks	
Text-to-Speech	text	*[espnet/kan-bayashi_ljspeech_vits, ...]*	*["his model was trained by kan-bayashi using ljspeech/tts1 recipe in...", ...]*
Audio-CLS	audio	*[TalTechNLP/voxlingua107-epaca-tdnn, ...]*	*["This is a spoken language recognition model trained on the...", ...]*
ASR	audio	*[jonatasgrosman/wav2vec2-large-xlsr-53-english, ...]*	*["Fine-tuned XLSR-53 large model for speech recognition in English ...", ...]*
Audio-to-Audio	audio	*[speechbrain/metricgan-plus-voicebank, ...]*	*["MetricGAN-trained model for Enhancement...", ...]*

DEMONSTRATION-BASED PARSING

- To better understand the intention and criteria for task planning, HuggingGPT incorporates multiple demonstrations in the prompt.
- Each demonstration consists of a user request and its corresponding output, which represents the expected sequence of parsed tasks.
- By incorporating dependencies among tasks, these demonstrations aid HuggingGPT in understanding the logical connections between tasks, facilitating accurate determination of execution order and identification of resource dependencies.

MODEL SELECTION

- Following task planning, HuggingGPT proceeds to the task of matching tasks with models, i.e., selecting the most appropriate model for each task in the parsed task list.
- To this end, they use model descriptions as the language interface to connect each model.
- They first filter out models based on their task type to select the ones that match the current task.
- Among these selected models, they rank them based on the number of downloads on Hugging Face and then select the top-K models as the candidates.

TASK EXECUTION

- In this stage, HuggingGPT will automatically feed these task arguments into the models, execute these models to obtain the inference results, and then send them back to the LLM.
- It is necessary to emphasize the issue of resource dependencies.
- To address of this issue, HuggingGPT employs a unique symbol.
- This symbol is utilized to track resources generated by depended tasks, identified as `<resource>-task_id`, where `task_id` represents the id of the depended task.

TASK EXECUTION

- During the task planning stage, if some tasks are dependent on the outputs of previously executed tasks (e.g., `task_id`), HuggingGPT sets this symbol (i.e., `<resource>-task_id`) to the corresponding resource subfield in the arguments.
- Then in the task execution stage, HuggingGPT dynamically replaces this symbol with the resource generated by the depended task.
- Furthermore, for the remaining tasks without any resource dependencies, they will execute these tasks directly in parallel to further improve inference efficiency.

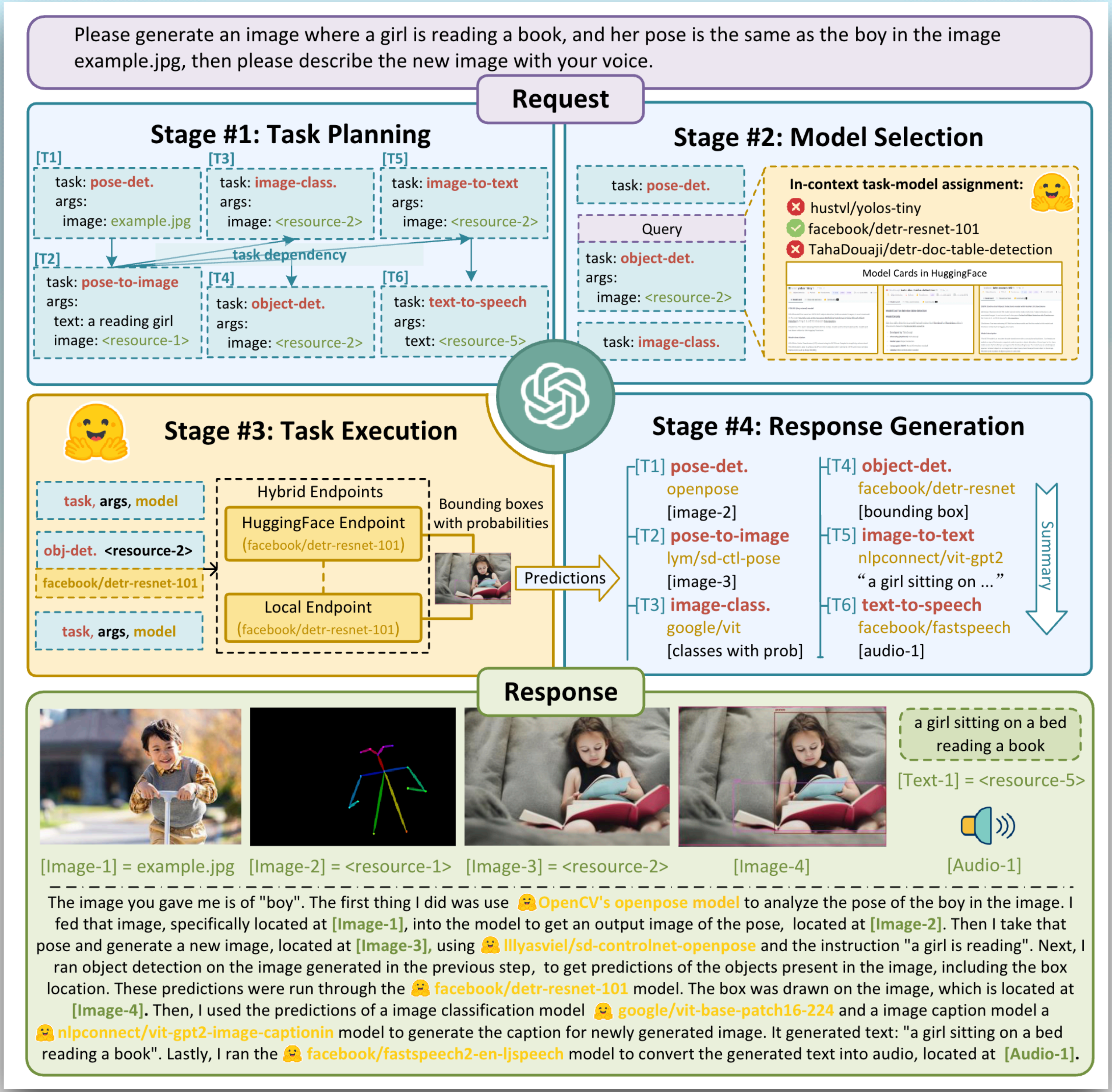
RESPONSE GENERATION

- After all task executions are completed, HuggingGPT needs to generate the final responses.
- HuggingGPT integrates all the information from the previous three stages (task planning, model selection, and task execution) into a brief summary in this stage, including the list of planned tasks, the selected models for the tasks, and the inference results of the models.
- Most important among them are the inference results, which are the key points for HuggingGPT to make the final decisions.

RESPONSE GENERATION

- These inference results are presented in a structured format, such as bounding boxes with detection probabilities in the object detection model, answer distributions in the question-answering model, etc.
- HuggingGPT allows LLM to receive these structured inference results as input and generate responses in the form of friendly human language.
- Moreover, instead of simply aggregating the results, LLM generates responses that actively respond to user requests, providing a reliable decision with a confidence level.

SUMMARY OF MAIN STAGES



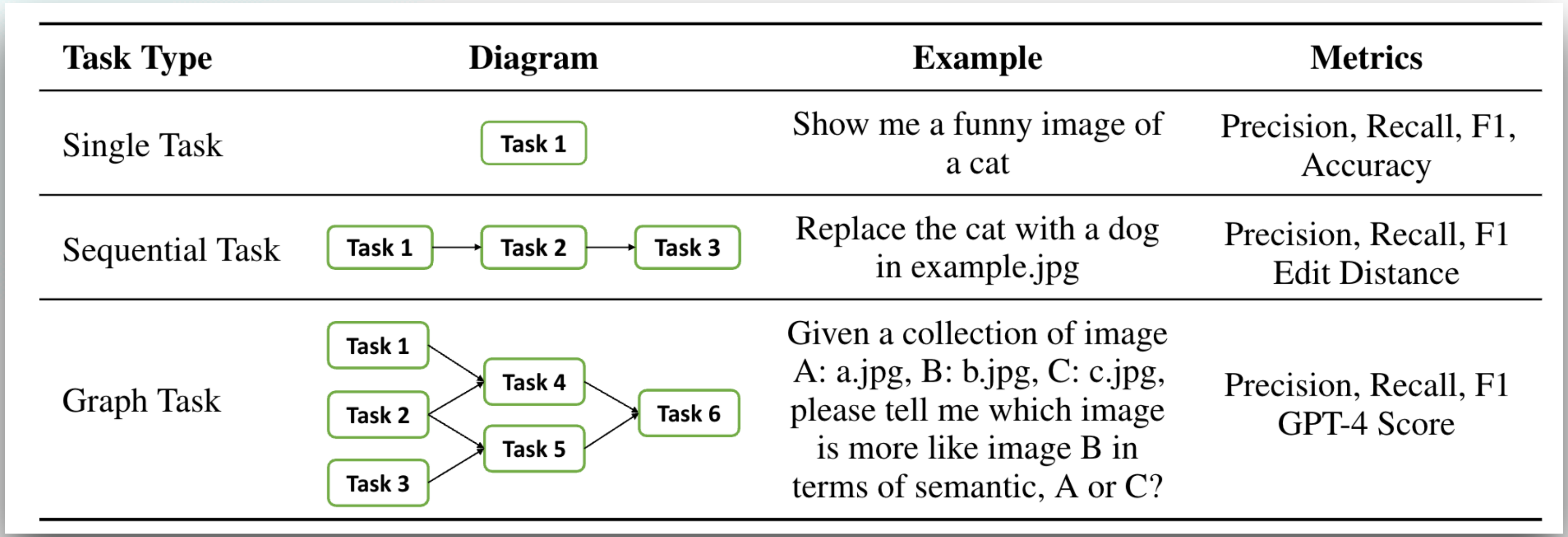
Please generate an image where a girl is reading a book, and her pose is the same as the boy in the image example.jpg, then please describe the new image with your voice.

Request
Stage #1: Task Planning
[T1] task: pose-det. args: image: example.jpg
[T3] task: image-class. args: image: <resource-2>
[T5] task: image-to-text args: image: <resource-2>
task dependency
[T2] task: pose-to-image args: text: a reading girl image: <resource-1>
[T4] task: object-det. args: image: <resource-2>
[T6] task: text-to-speech args: text: <resource-5>
Stage #2: Model Selection
task: pose-det.
Query
task: object-det. args: image: <resource-2>
task: image-class.
In-context task-model assignment:
hustvl/yolos-tiny
facebook/detr-resnet-101
TahaDouaji/detr-doc-table-detection
Model Cards in HuggingFace
Stage #3: Task Execution
task, args, model
obj-det. <resource-2>
facebook/detr-resnet-101
task, args, model
Hybrid Endpoints
HuggingFace Endpoint (facebook/detr-resnet-101)
Local Endpoint (facebook/detr-resnet-101)
Bounding boxes with probabilities
Predictions
Stage #4: Response Generation
[T1] pose-det. openpose [image-2]
[T2] pose-to-image lym/sd-ctl-pose [image-3]
[T3] image-class. google/vit [classes with prob]
[T4] object-det. facebook/detr-resnet [bounding box]
[T5] image-to-text nlpconnect/vit-gpt2 "a girl sitting on ..."
[T6] text-to-speech facebook/fastspeech [audio-1]
Summary
Response
a girl sitting on a bed reading a book
[Text-1] = <resource-5>
[Image-1] = example.jpg
[Image-2] = <resource-1>
[Image-3] = <resource-2>
[Image-4]
[Audio-1]

The image you gave me is of "boy". The first thing I did was use 🤗 OpenCV's openpose model to analyze the pose of the boy in the image. I fed that image, specifically located at [Image-1], into the model to get an output image of the pose, located at [Image-2]. Then I take that pose and generate a new image, located at [Image-3], using 🤗 lllyasviel/sd-controlnet-openpose and the instruction "a girl is reading". Next, I ran object detection on the image generated in the previous step, to get predictions of the objects present in the image, including the box location. These predictions were run through the 🤗 facebook/detr-resnet-101 model. The box was drawn on the image, which is located at [Image-4]. Then, I used the predictions of a image classification model 🤗 google/vit-base-patch16-224 and a image caption model a 🤗 nlpconnect/vit-gpt2-image-captionin model to generate the caption for newly generated image. It generated text: "a girl sitting on a bed reading a book". Lastly, I ran the 🤗 facebook/fastspeech2-en-ljspeech model to convert the generated text into audio, located at [Audio-1].

EXPERIMENTS & RESULTS

❖ To conduct their evaluation, they invite some annotators to submit some requests. Then they collect these data as the evaluation dataset. They use GPT-4 to generate task planning as the pseudo labels, which cover single, sequential, and graph tasks.

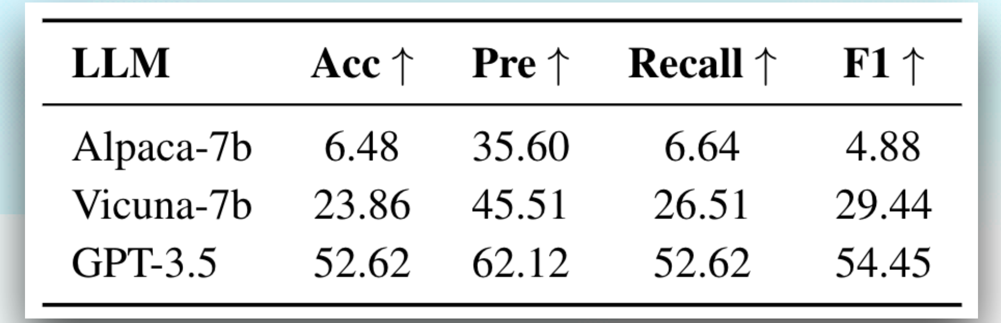


Task Type	Diagram	Example	Metrics
Single Task	Task 1	Show me a funny image of a cat	Precision, Recall, F1, Accuracy
Sequential Task	$\text{Task 1} \rightarrow \text{Task 2} \rightarrow \text{Task 3}$	Replace the cat with a dog in example.jpg	Precision, Recall, F1 Edit Distance
Graph Task	$\text{Task 1} \rightarrow \text{Task 4}$; $\text{Task 2} \rightarrow \text{Task 4}$; $\text{Task 2} \rightarrow \text{Task 5}$; $\text{Task 3} \rightarrow \text{Task 5}$; $\text{Task 4} \rightarrow \text{Task 6}$; $\text{Task 5} \rightarrow \text{Task 6}$	Given a collection of image A: a.jpg, B: b.jpg, C: c.jpg, please tell me which image is more like image B in terms of semantic, A or C?	Precision, Recall, F1 GPT-4 Score

Evaluation for task planning in different task types

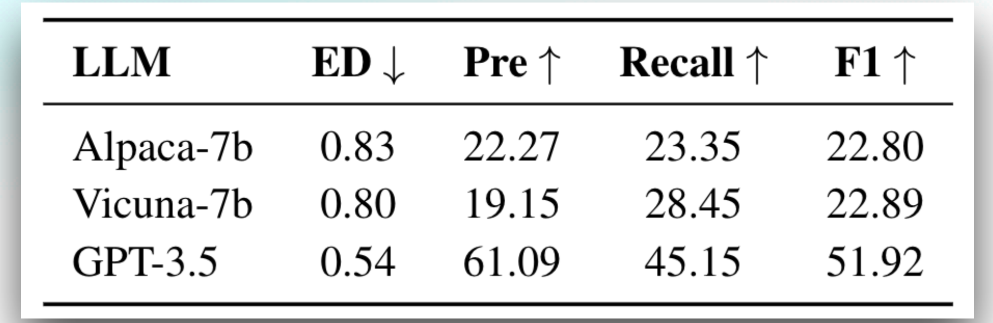
EXPERIMENTS & RESULTS

- The following tables show the planning capabilities of HuggingGPT on the three categories of GPT-4 annotated datasets, respectively.



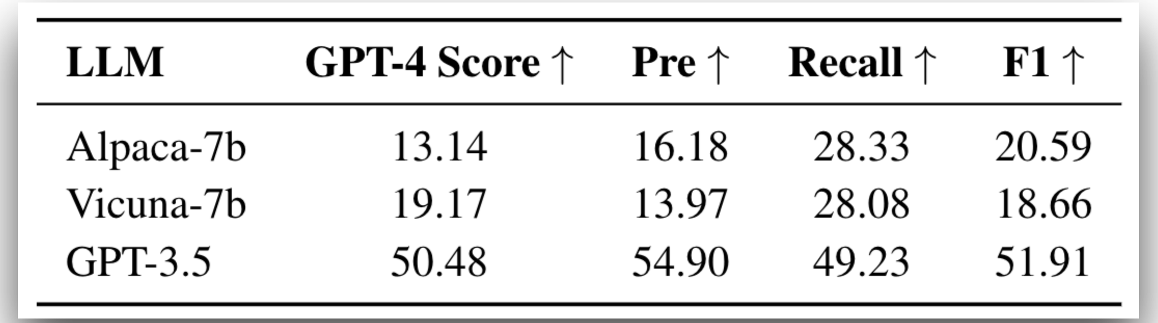
LLM	Acc ↑	Pre ↑	Recall ↑	F1 ↑
Alpaca-7b	6.48	35.60	6.64	4.88
Vicuna-7b	23.86	45.51	26.51	29.44
GPT-3.5	52.62	62.12	52.62	54.45

Evaluation for the single task. “Acc” and “Pre” represents Accuracy and Precision



LLM	ED ↓	Pre ↑	Recall ↑	F1 ↑
Alpaca-7b	0.83	22.27	23.35	22.80
Vicuna-7b	0.80	19.15	28.45	22.89
GPT-3.5	0.54	61.09	45.15	51.92

Evaluation for the sequential task. “ED” means Edit Distance

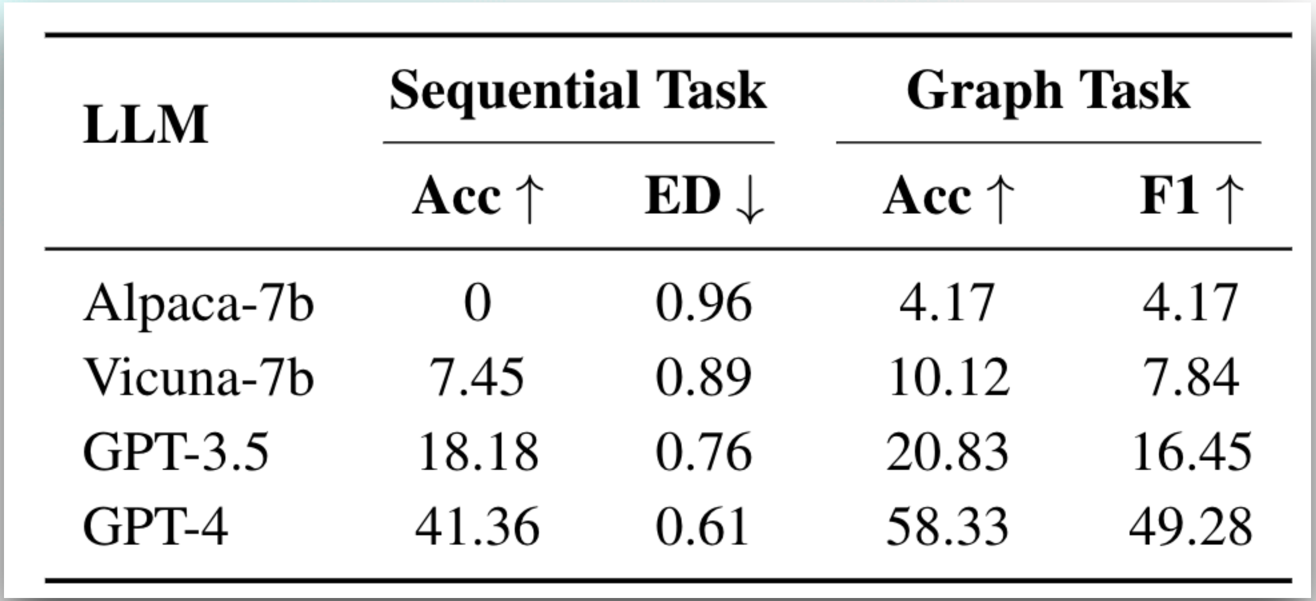


LLM	GPT-4 Score ↑	Pre ↑	Recall ↑	F1 ↑
Alpaca-7b	13.14	16.18	28.33	20.59
Vicuna-7b	19.17	13.97	28.08	18.66
GPT-3.5	50.48	54.90	49.23	51.91

Evaluation for the graph task

EXPERIMENTS & RESULTS

- Furthermore, they invite some expert annotators to label task planning (Instead of using GPT-4) for some complex requests (46 examples) as a high-quality human annotated dataset.



LLM	Sequential Task		Graph Task	
	Acc ↑	ED ↓	Acc ↑	F1 ↑
Alpaca-7b	0	0.96	4.17	4.17
Vicuna-7b	7.45	0.89	10.12	7.84
GPT-3.5	18.18	0.76	20.83	16.45
GPT-4	41.36	0.61	58.33	49.28

Evaluation on the human-annotated dataset

EXPERIMENTS & RESULTS

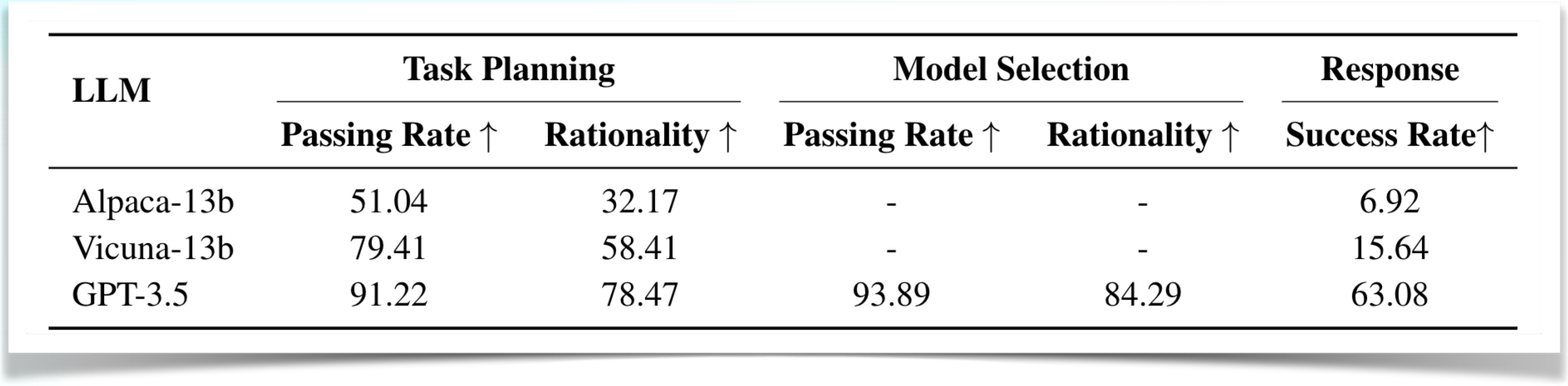
- In addition to objective evaluations, they also invited three human experts to conduct a subjective evaluation in their experiments.
- They collected 130 diverse requests to evaluate the performance of HuggingGPT at various stages, including task planning, model selection, and final response generation.
- Three human experts were asked to annotate the provided data according to the well-designed metrics and then calculated the average values to obtain the final scores.

EXPERIMENTS & RESULTS

- They designed three evaluation metrics, namely passing rate, rationality, and success rate.
- **Passing Rate:** to determine whether the planned task graph or selected model can be successfully executed.
- **Rationality:** to assess whether the generated task sequence or selected tools align with user requests in a rational manner.
- **Success Rate:** to verify if the final results satisfy the user’s request.

EXPERIMENTS & RESULTS

- These results indicate that their objective evaluations are aligned with human evaluation and further demonstrate the necessity of a powerful LLM as a controller in the framework of autonomous agents.



LLM	Task Planning		Model Selection		Response
	Passing Rate ↑	Rationality ↑	Passing Rate ↑	Rationality ↑	Success Rate↑
Alpaca-13b	51.04	32.17	-	-	6.92
Vicuna-13b	79.41	58.41	-	-	15.64
GPT-3.5	91.22	78.47	93.89	84.29	63.08

Human Evaluation on different LLMs. They report two metrics, passing rate (%) and rationality (%), in the task planning and model selection stages and report a straightforward success rate (%) to evaluate whether the request raised by the user is finally resolved.

LIMITATIONS

❖ **Planning:** In HuggingGPT, task planning heavily relies on the capability of LLM. Consequently, we cannot ensure that the generated plan will always be feasible and optimal.

❖ **Efficiency:** HuggingGPT requires multiple interactions with LLMs throughout the whole workflow and thus brings increasing time costs for generating the response.

❖ **Instability:** This is mainly caused because LLMs are usually uncontrollable. Although LLM is skilled in generation, it still possibly fails to conform to instructions or give incorrect answers during the prediction, leading to exceptions in the program workflow.

Thank you for your attention